Some comments on Finding a Median

We will actually try to find the element of rank R for any rank R. We will use the notation x<y when x ranks lower than y, though that might be considered the opposite of usual notation.

Suppose we choose to do so by dividing our keys into blocks of size 5 (with perhaps a remainder) sorting each of these, and using the median of their middle elements as a comparison key.

If it takes f(N) comparisons to find the hardest to find rank R(N) we can find the following recursion inequality

f(N) <

the number of steps needed to sort the various 5's separately

+ the
number of steps needed to find the median, x, of the middle elements of the 5's

+ the
number of steps needed to find whether that median ranks higher or lower than R

+ the
number of steps needed to find rank R among those keys on the same side as R from that median.

This yields us at worst

$$f(N) \le 7N/5 + f(N/5) + 2N/5 + f(7N/10),$$

since we can completely sort each 5-tuple with 7 comparisons, we can find the median of the N/5 middles of the 5-tuples with f(N/5) comparisons, can compare that median with the remaining keys with 2N/5 comparisons to find its exact rank, and can then eliminate 3N/10 keys that are on the opposite side of the median from R and find the key at appropriate rank in among the 7N/10 remaining keys with f(7N/10) comparisons..

And we can massage this and an appropriate induction hypothesis to get a bound on f(N).

We can improve this bound by noticing that once we complete the third of these steps, if we only eliminate those keys that we know from the second step are on the side opposite to x from rank R, we will end up with roughly N/10 sets of 5 keys that are each still sorted and the same number of sets of two sorted keys.

To apply the last step starting with this, we do not have to do the first step above in its entirety; some of our keys will still be sorted 5-tuples; we only have to regroup the B/10 pairs we have into 5s. This will take at most 5 comparisons for each 5 produced (since the ordered pairs give us already the first two comparisons out of 7 needed to sort 5s) and there will be N/5 keys that have to be regrouped, which means only N/5 comparisons will be needed to reach the second step on the 7N/10 remaining keys, instead of 49/50N of them that we would have to perform if we had started from scratch in sorting into 5-tuples.

This gives us

$$f(N) \le 7N/5 + f(N/5) + 2N/5 + f(7N/10) - (49/50)N + N/5$$

$$f(N) \leq (51/50)N + f(N/5) + f(7N/10).$$

Before continuing to try to improve this inequality, let us notice how to apply it.

We make the assumption that for all values of M less than N, we have $f(M) < cM$. (We do not know yet what c is but so what?)

Then if we apply this assumption on the right here we get

$$F(N) \leq 51N/50 + 9c/10.$$

This means that if we have $c \geq 51/5$ (or $c \geq 10.2$) then $51N/50$ is less than $c/10$ and we can conclude (by substituting this statement into the last equation, that $F(N) \leq cN$.

Now we try to improve this bound.

The third step involves comparing our median middle x with 2 ordered keys out of each sorted 5. We want to make use of the ordering of these keys to reduce our bound on the number of comparisons.

Suppose in performing the third step we keep track of three numbers as we do so..

1. The lowest possible rank for x; call it L
2. The rank R that we are seeking
3. The highest possible rank for our x: call it H

At the beginning of step three these are (roughly) $L=3N/10$, $R=R$, $H=7N/10$. If R is not between L and H we can omit step 3 entirely; for example, if R is greater than H nothing below L can be R and we can eliminate all keys below L.

We want to find how x ranks in comparison to the two keys in each 5-tuple that we do not already know this about. When some key is found to be less than x ,
we can increase L by 1, and when the key is greater than x we can lower H by 1. We can stop step 3 when we find either H=R or L=R. If we find H=R we know that our median x is at or below R and anything below x cannot be R. Thus everything that is L or below can be eliminated.

Suppose we compare x with the lower of the two relevant keys in a given 5-tuple. That is, if the 2 keys are y and z and $y< z$ holds, suppose we compare x with y. Then if we find $x<y$ then we decrease H by 2 with only one comparison.

We want to make use of this fact to improve our estimates. To do so we keep track of which of N-H and L is smaller. If L is smaller than N-H, we compare x
with y. Then, if x is less than y we find out by transitivity that x is less than z as well, and then H decreases by 2 from only one comparison.
If x is greater than y then L increases by only 1, but that is good for us because L was smaller than N-H. We go on and compare x with the appropriate key from the next 5-tuple .

We when N-H is smaller than L we compare x with z first with similar conclusions,

When L and N-H are equal, the one that increases by 1 will necessarily become the bigger of the two, and the smaller will stay the same.

.Suppose, in applying all this we compare x once in each of A 5-tuples, encounter B ties between L and N-H and in C comparisons other than ties the smaller increases by 1 in the comparison. (We suppose that in the case of ties we always have bad luck and the larger after our comparison never increases by 2 from it.)

Then the bigger of the two will increase by 2A-2C-B, and the smaller will increase by C. By definition, 2A-2C-B is greater than C.

If at the end of these comparisons we have L bigger than N-H, we find L will have grown to 3N/10+2A-2C-B, and H will have descended to 7N/10-C.

We also know that B cannot exceed C+1, since there cannot be two ties at the same value of H. Se we assume the worst, that B=C+1,. We find that after A
comparisons, we have increased L to 3N/10+ 2A-3C-1 and H will have decreased to 7N/10 − C.

Notice that, assuming the worst, that step 3 ends with R=L, the task remaining in step 4 will be f(7N/10 − C). We then have R=3N/10+2A -3C-1.

These statements allow us to determine the worst case number of comparisons needed to perform steps 3 and 4 as a function of C ,
R and N, but we also have to consider the increased cost of reforming 5-tuples after step 3. It seems that this cost can be held to 2C.

Putting all this together we get as a bound on the number of steps of

For R at least N/2, of

F(N) < 7N/5 + f(N/5) + (R-3N/10)/2 +f(7N/10-C) − 49N/50 +N/5 +7C/5 +3C +2C.

Finally, notice that after performing these steps once, the worst case performance of these steps means that in dealing with the 7N/10 − C remaining keys in step 4 there will be no step 3. That is, if we start off with R=N/2, which means we are seeking the minimum, after we eliminate at least 3N/10 keys on one side, we will have M keys left with M at most 7N/10 are looking for a rank of 2M/7 or less (or M- that number), But 2M/7 is less than 3M/10, so that no step 3 will be necessary next time.

In fact, if R lies between 3N/10 and 7N/10, after lopping off from the side that R is closer to will put R outside that range on the next iteration, On the other hand, if we lop off from the further side, so that the R in the next iteration is not outside that range, step 3 will take fewer steps, and we can bound (R-3N/10) above by N/5.

When C is 0, we get the worse of two possible bounds: The first applies if youi eliminate keys so that the reduced problem has an R that lies between 3M/10 and 7M/10 and the second applies otherwise. In the second the reduction is iterated.

$$F(N) < 36N/50 + f(N/5) + f(7N/10)$$

$$\text{and } F(N) < 41N/50 + f(N/5) + f(7N/50) + f(49N/100) + (31/50)*(7N/10)$$

I seem to get something like F(N) < 8.2N, (but this is probably wrong.)

By taking derivatives of the expressions above you can show that the worst case occurs when C=0.